Šablonovací systém pro správu a generování emailů

autor: Josef Kumstýř | vedoucí: Mgr. Pavel Ježek, Ph.D. | 2019

Cíle práce

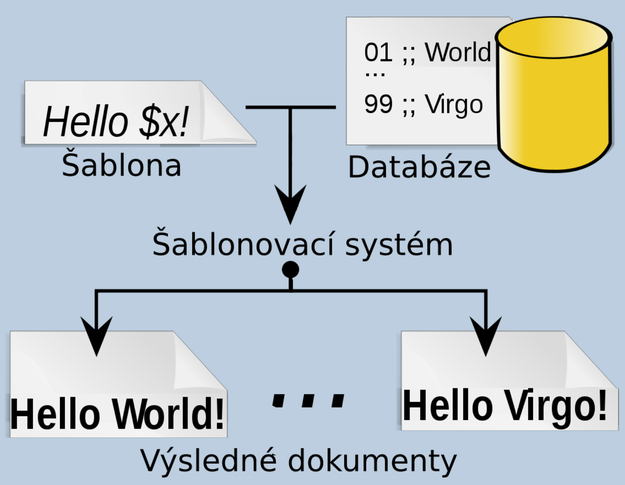
Cílem této práce bylo vytvořit šablonovací systém, který nahradí a vylepší stávající šablonovací systém používaný v e-shopu Alza.cz.

Účelem tohoto systému je správa šablon a generování zpráv (HTML emailů, SMS, apod.) z těchto šablon. Součástí zadání byla nutnost zachovat funkčnost stávajících šablon z původního systému.

Co je šablonovací systém?

Na obrázku 1 vidíme princip fungování šablonovacího systému. Prvním argumentem je *šablona*, která na obrázku obsahuje text „Hello $x!“. Druhým argumentem je *databáze* obsahující potřebná data k sestavení šablony.

Sestavením šablony rozumíme vygenerování výsledného dokumentu z textu šablony použitím konkrétních dat z databáze. Na obrázku mají výsledné dokumenty text „Hello World!“ a „Hello Virgo!“, v závislosti na použitých datech. Výsledný dokument může být v libovolném formátu (HTML, JSON, prostý text, ...).



Obrázek 1: Princip šablonovacího systému

Původní systém

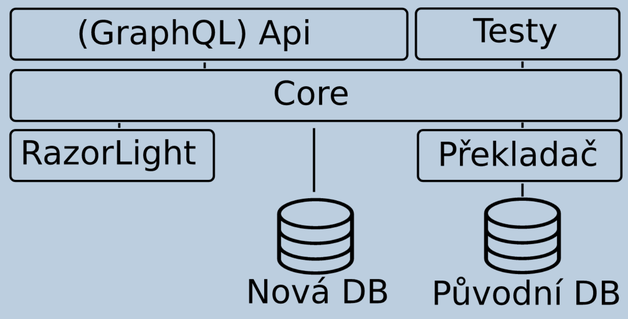
Původní systém je šablonovací systém, text jeho šablon se skládá ze tří elementů. Prvním je obyčejný text, který je při sestavení bez úpravy zkopírován do výsledného dokumentu. Druhým jsou klíčová slova, která jsou zpracována podle dokumentace a mohou vygenerovat text, který je vložen na jejich místo do výsledného dokumentu. Třetím elementem jsou formátovací značky, které upravují text vytvořený klíčovým slovem.

Na obrázku 2 vidíme příklad šablony, která vypíše zítřejší datum. „Date“ je klíčové slovo, které s parametrem 1 vypíše zítřejší datum. „Format“ je formátovací značka, která upraví vrácené datum do zvoleného formátu. Výsledkem může být řetězec „Zítra je 06.09“.

```
Zítra je {Date.1:Format.DD.MM}
```

Obrázek 2: Princip šablonovacího systému

Architektura



Obrázek 3: Architektura systému

Core

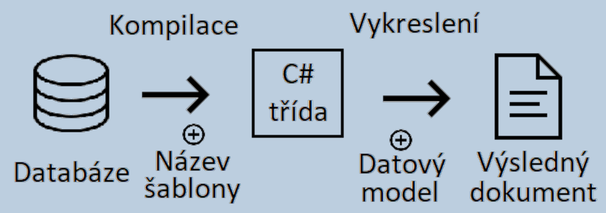
Projekt *Core* zajišťuje:

- Rozšiřitelnost jazyka šablon - všechny šablony mají za předka společnou třídu, která poskytuje metody (např. *Include*, nebo *Query* v příkladu níže) použitelné v kódu šablon a je jednoduše rozšiřitelná.
- Správu šablon - ukládání a upravování.
- Cachování zkompilovaných šablon - zkompilování šablony (viz *RazorLight*) trvá cca 50x déle než vykreslení již zkompilované šablony. Proto zkompilované šablony ukládáme do cache.
- Varianty šablon - každá šablona může mít různé varianty s odlišnými texty. V kódu pak můžeme dynamicky (na základě dat z db) nastavit, která varianta se pro konkrétní sestavení použije.
- Náhled šablon - při testování šablony nevíme, s jakými argumenty bude zavolána. Proto má každá šablona svoje testovací argumenty a lze ji s nimi sestavit.
- A jiné...

RazorLight

Po analýze původního systému jsme dospěli k závěru, že není výhodné používat vlastní syntaxi. Důvodem je časová i ekonomická náročnost rozšiřování a údržby vzhledem k tomu, kolik je existujících kvalitních řešení.

Pro náš systém jsme použili knihovnu *RazorLight*, jejíž princip fungování je popsán na obrázku 4. Text šablony uložený v db je zkompilován do C# třídy, která je poté použita k vykreslení šablony. Tato knihovna používá známou *Razor* syntaxi, která vychází z jazyka C# a je používána frameworkem ASP.NET.



Obrázek 4: Princip sestavení šablony

Překladač

Protože jedním z cílů práce bylo zachovat funkčnost existujících šablon, tak jsme v rámci práce vytvořili překladač šablon ze staré syntaxe do nové. Princip fungování překladače vidíme na obrázku 5. Pro každé klíčové slovo jsme vytvořili ekvivalentní C# funkci, na kterou je slovo přeloženo.

Obrázek 5: Fungování překladu

Srovnání systémů

```
// Vlož HTML hlavičku a začni tělo
@Include("EmailStart")

// Vypiš "Hello World"
@{var name = "World"}
<h1>Hello @Name</h1>

// Najdi uživatele v databázi
@{var user = Query("ProcedureName")}

// Zobraz jméno pokud je uživ. dospělý
@if((int)user["age"] >= 18)
  <p>@user["name"] is adult!</p>

@Include("EmailEnd")
```

```
{// Vlož HTML hlavičku a začni tělo}
{Include."EmailStart"}

{// Vypiš "Hello World"}
{Declare.Var,Name,"World"}
<h1>Hello {Var.Name}</h1>

{// Najdi uživatele v databázi}
{Query.User,"ProcedureName"}

{// Zobraz jméno pokud je dospělý}
{// Tělo podmínky musí mít vlastní šablonu}
{// Db musí spočítat výsledek podmínky}
{Include.ShowUserName%,User.IsAdult}

{Include."EmailEnd"}
```

Obrázek 6: Nová Razor syntaxe (vlevo); původní vlastní syntaxe (vpravo)

Na obrázku 6 vidíme ukázku textu stejné šablony v nové a ve staré syntaxi, kde si můžeme všimnout zjednodušení podmíněného zobrazení. Výsledný dokument těchto šablon vidíme na obrázku 7. Nejvýznamnější vylepšení oproti původnímu systému jsou:

1. Nové možnosti jazyka (@if, @while, celý jazyk C#)
2. Rozšiřitelnost jazyka (pomocí společných metod - viz *Core*)
3. Přidání lokálních proměnných, možnost jednoslovných názvů

Hello World

Peter is adult!

Obrázek 7: Výsledek sestavení šablony z obr. 6

Poděkování

Děkuji Mgr. Pavlu Ježkovi, Ph.D. za jeho cenné rady a pomoc při psaní této práce a kolegům z firmy *Alza.cz* za jejich pomoc a umožnění spojení tohoto projektu s bakalářskou prací.

Závěr

Náš systém splňuje všechny stanovené cíle. V rámci firmy *Alza.cz* je vytvořený systém momentálně používán pro posílání veškeré SMS komunikace. Současně je testován na šablonách sestavující emaily a brzy tedy bude používán k sestavování všech zpráv zákazníkům tohoto e-shopu.